ISSN 1561-5359. Штучний інтелект, 2016, № 2

УДК 004.932

П.Ю. Сабельніков
Інститут кібернетики імені В.М. Глушкова НАН України,
просп. Академіка Глушкова, 40, м. Київ-187, Україна, 03680

ПАРАЛЕЛЬНЕ КОДУВАННЯ КОНТУРІВ ОБ'ЄКТІВ У БІНАРНИХ ЗОБРАЖЕННЯХ ТА ОБЧИСЛЕННЯ ЇХ МОМЕНТІВ ІНЕРЦІЇ

P.Yu. Sabelnikov

V.M. Glushkov Institute of Cybernetics of the National Academy of Sciences of Ukraine
Academik Glushkov av., 40, c. Kyiv-187, Ukraine, 03680

PARALLEL ENCODING THE CONTOURS OF OBJECTS IN THE BINARY IMAGE AND THE CALCULATION OF ITS MOMENTS OF INERTIA

У статті пропонуються алгоритми кодування контурів об'єктів у бінарних зображеннях та обчислення їх моментів інерції, які можуть значно прискорити процес обчислень за рахунок суміщення введення і обробки відеоданих з використанням векторних операцій.

Ключові слова: зображення, контур, моменти інерції.

The article proposes coding algorithms contours of objects in the binary images, and calculating their moments of inertia, which can greatly speed up the calculation process at the expense of combining input and processing of image data using vector operations.

Key words: images, contour, moments of inertia.

Вступ

Достатньо важливою при обробці зображень, особливо вбудованими пристроями, що призначені для роботи в реальному часі, є проблема прискорення обчислень. У даній статті пропонуються алгоритми кодування контурів об'єктів у бінарних зображеннях та обчислення їх моментів інерції, які можуть значно прискорити процес обчислень за рахунок суміщення введення і обробки відеоданих з використанням векторних операцій.

Процедура послідовного виділення і кодування контурів передбачає наявність в пам'яті обчислювального пристрою повного кадру зображення. З метою скорочення часу запізнювання результатів обробки, економії ресурсів пам'яті та підвищення швидкодії за рахунок використання тільки внутрішньої швидкої пам'яті процесора, доцільно проводити обчислення паралельно з введенням зображення, послідовно отримуючи і зберігаючи тільки частку його рядків. Цій меті відповідає метод виділення і кодування контурів об'єктів бінарних відеозображень, заснований на послідовному скануванні частки рядків зображення, представлений в [1].

Метод, у загальному плані, має наступні основні етапи.

Вводиться черговий i-ий рядок зображення і бінаризується. Проводиться послідовний аналіз взаємного розташування чорних і білих сегментів в i-ому та (i-1) - ому рядках, при цьому можуть виникати 4 варіанти обставин:

1. В обох рядках закінчились чорні сегменти або їх зовсім не було, тоді повертаємося до введення наступного рядка (i=i+1).

2. В i - ому рядку зустрівся чорний сегмент, на тлі білого сегменту (i-1) - ого рядка, або навпаки, білий сегмент на тлі чорного сегменту. Така ситуація відповідає зародженню двох гілок контуру, яким призначаються чергові порядкові номери. Кінцевий номер цих гілок, а саме якому контуру вони належать, стане відомий в кінці процесу після з'єднання гілок всіх контурів.

3. В (i-1) - ому рядку зустрівся чорний сегмент, на тлі білого сегменту (i) - ого рядка або навпаки, білий сегмент, на тлі чорного сегменту. Така ситуація відповідає з'єднанню двох гілок. При цьому обом гілкам призначається менший з двох їх номерів, якщо вони різні.

4. В i - ому рядку зустрівся чорний сегмент, що перетинається з чорним сегментом (i-1) - ого рядка. Згідно з порядком слідування пікселів у рядках можливі наступні варіанти. За початком чорного сегменту i - ого рядка йде початок чорного сегменту (i-1) - ого рядка або навпаки, за початком чорного сегменту (i-1) - ого рядка йде початок чорного сегменту i - ого рядка. Такі варіанти можливі і з початками білих сегментів. Кожен з варіантів означає перехід гілки з (i-1) - ого рядка в (i) - ий рядок.

У результаті послідовного аналізу всього бінарного зображення формуються вектори контурів всіх обєктів зображення, що обєднують координати точок гілок, які належать кожному з контурів.

Алгоритм паралельного кодування контурів із застосуванням векторних операцій

Для подальшого прискорення процесу локалізації і кодування контурів об'єктів пропонується модифікація запропонованого методу і алгоритм обробки бінарних відеозображень із застосуванням векторних операцій.

У рамках роботи будемо розглядати бінарне зображення, у якому на білому фоні представлені чорні об'єкти (або, навпаки, це не принципово), що мають замкнені контури і позначені як об'єкти 1-ого рівня. При цьому, в межах об'єктів 1-ого рівня можуть бути розташовані білі об'єкти 2-ого рівня, в межах об'єктів 2-ого рівня можуть бути розташовані чорні об'єкти 3-ого рівня і так далі. Контур визначається як замкнена ломана лінія товщиною в один піксель, яка з'єднує центри граничних пікселів об'єкта, що примикають друг до друга, по прямій або діагоналі і належать цьому об'єкту. Щоб не виникало колізій з перетинанням контурів, встановлено, що точка між пікселями при переходах по діагоналі належить чорним об'єктам і є перепоною для білих об'єктів.

Контури представляються у вигляді таблиці 1, компонентами якої, є:

n – номери гілок (після закінчення аналізу будуть змінені на номери контурів),

N – загальна кількість гілок (буде відома в кінці аналізу),

p – ознака гілки (1 – чорна гілка, 0 – біла гілка),

y – номер (координата) рядка, з якого починається гілка,

z – кількість точок у гілці,

x – номери (координати) точок гілки в рядку,

n_p – номер гілки, з кінцем якої здійснюється з'єднання.

Таблиця 1. Таблиця гілок контурів

n	*p*	*y*	*z*	*x*	*n_p*
0					
1					
2					
3					
………………					
N					
N+1					

Пояснімо роботу формальних правил формування таблиці гілок контурів на прикладі тестового бінарного зображення, представленого на рисунку 1. Слід враховувати, що, для коректності роботи алгоритму, об'єкти не повинні стикатися з краями зображення. Для довільного зображення крайні рядки і стовпці заповнюються фоном.

Для подальших пояснень введемо позначення початків чорного і білого сегментів бінарного зображення:

1 – початок чорного сегмента (*i*-1) - ого рядка,

$\underline{1}$ – початок чорного сегмента (*i*) - ого рядка,

0 – початок білого сегмента (*i*-1) - ого рядка,

$\underline{0}$ – початок білого сегмента (*i*) - ого рядка.

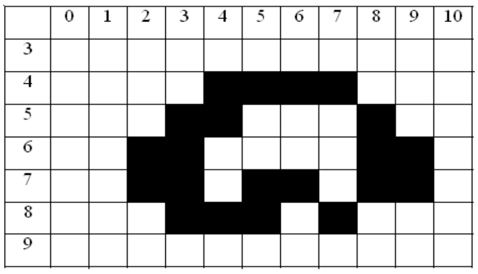


Рис. 1. Тестове бінарне зображення.

Згідно з методом сканування рядків зображення, можливі ситуації при послідовному аналізі, що відповідають наступним комбінаціям слідування початків сегментів за порядковими номерами точок попереднього і поточного рядків:

1. ($\underline{1}$, 1, …), (1, $\underline{1}$, …), ($\underline{0}$, 0, …), (0, $\underline{0}$, …) – продовження гілок з рядка в рядок;

2. (0, $\underline{…}$, **$\underline{1}$, $\underline{0}$**, $\underline{...}$, *), (1, $\underline{…}$, **$\underline{0}$, $\underline{1}$**, $\underline{...}$, *) – виділені жирним шрифтом комбінації відповідають зародженню пари з'єднаних початками чорної і білої гілок у поточному рядку (зірочка позначає перший не підкреслений індекс, що зустрінеться);

3. ($\underline{0}$, …, **1, 0**, ..., $\underline{*}$), ($\underline{1}$, …, **0, 1**, ..., $\underline{*}$) – виділені жирним шрифтом комбінації відповідають з'єднанню кінців пари гілок у попередньому рядку, що означає приписування до параметрів кожної гілки номеру гілки, з якою вона з'єднується (підкреслена зірочка позначає перший підкреслений індекс, що зустрінеться);

4. Для інших сполучень трьох індексів продовжити аналіз.

Згідно з умовою, що точка між чорними пікселями при переходах по діагоналі належить чорним об'єктам, при одночасній зустрічі в одній позиції комбінацій 0 і 1 першою вважається одиниця.

Для формального представлення алгоритму введемо деякі поняття і позначення.

Поточний рядок при перетворенні в бінарну форму будемо представляти вектором, де кожній точці відповідають 2 біта. Нуль відповідає білій точці, а одиниця – чорній. Для поточного рядка формуємо вектор *T* початків сегментів бінарного зображення в рядку наступним чином.

У векторі *T* формуємо бінарне зображення, наприклад, 4-ого рядка.

T=00,00,00,00,01,01,01,01,00,00,00.

Здійснюємо перетворення $T = T \cap (T\rangle\rangle 2)$, де

$\cap$ – операція покомпонентного логічного множення,

$\rangle\rangle 2$ – операція зсуву вектора на 2 біта вправо.

У результаті отримуємо *T*=00,00,00,00,01,00,00,00,01,00,00.

До результуючого вектору *R* поміщаємо вектор *T*, зсунутий на 1 біт вліво ($R = T\langle\langle 1$), і вважаємо його як вектор, сформований для попереднього рядка. Потім таким же чином формуємо в векторі *T* поточний 5-ий рядок та операцією логічного додавання $\cup$ додаємо його до вектора *R*. У результаті отримуємо злиті та впорядковані за номерами точок початки сегментів попереднього і поточного рядків.

R=00,00,00,01,10,01,00,00,11,01,00.

Надалі, для простоти, будемо представляти результуючі вектори не в бітовій, а в кодовій формі, де

00=0 означає, що в цій позиції немає початків сегментів,

01=1 означає початок сегмента в (i) - ому рядку,

10=2 означає початок сегмента в (i-1) - ому рядку,

11=3 означає одночасний початок сегментів в (i) - ому та в (i-1) - ому рядках.

Отже, для 5-ого рядка R=0,0,0,1,2,1,0,0,3,1,0.

Початки сегментів кожного рядка обов'язково чергуються: за початком чорного сегмента йде початок білого сегмента, а за початком білого сегмента йде початок чорного сегмента. Тому, при послідовному аналізі *R,* буде відомо не тільки, в якому рядку (попередньому чи поточному) зустрівся початок сегмента, а і якого він кольору.

Таким чином, алгоритм включає наступні дії.

На початку аналізу i та вектор R у всіх позиціях дорівнюють 0.

1. Послідовно з (i) - ого рядка бінарного зображення формується вектор T.
2. $R = R \cup T$.
3. Аналіз R і заповнення таблиці гілок контурів.
4. $R = T\langle\langle 1$, i=i+1.
5. Дії з п.1 повторюються, доки не дійдемо до кінця зображення.
6. Об'єднання гілок у контури.

Алгоритм аналізу і формування таблиці гілок контурів

Аналіз вектора R можна здійснювати послідовно, позиція за позицією, або за рахунок визначення значущих позицій, використовуючи, наприклад, операції нормалізації. Операція нормалізації – це операція, при якій обчислюється номер позиції першої «1» зі зсувом даних до цієї позиції. Тобто, по черзі здійснюється нормалізація бінарного рядка до першої «1», що відповідає початку сегмента, з обчисленням його загального номера в рядку. Технічні рішення швидкої однотактної операції нормалізації відомі і реалізовані в багатьох сигнальних процесорах, що серійно випускаються. Наприклад, у процесорах фірми Analog Devices.

Реалізовувати алгоритм пропонується за допомогою автоматної моделі, що циклічно, залежно від значень окремих позицій вектора R, які подаються на вхід автомата, змінює свій стан та залежно від цього здійснюються необхідні дії. Таблиця переходів автомата представлена в таблиці 2. Стан автомата вказує, які дві попередні значущі позиції вектора R потрібно враховувати при аналізі значення наступної позиції цього вектора, що надходить на вхід автомата, а також, початок якого сегмента білого чи чорного буде закодовано в цій позиції. Тобто, коли стан позначено (0, $\underline{0}$), з кодом 0, це означає, що в попередніх позиціях поточного і попереднього рядків були початки білих сегментів. Відповідно, надалі в обох рядках можуть зустрітися тільки початки чорних сегментів. У стовпчику «Дії при зміні стану» таблиці 2 однією зірочкою позначені дії при зародженні гілок, двома – при з'єднанні гілок, трьома – при переході гілки з рядка на рядок.

У поясненнях задіяні такі проміжні змінні та вектори:

i – номер поточного рядка,

ss – лічильник номерів гілок (перед початком аналізу $ss=0,$ в кінці аналізу буде дорівнювати кількості гілок),

$stan$ – код стану автомата, $stan$ =0 перед початком аналізу кожного вектора $R,$

j – номер поточної позиції вектора R, $j=0$ перед початком аналізу кожного вектора R,

$vxid$ – код значення поточної позиції вектора R,

nn, n_min, n_max – проміжні значення номерів гілок,

zz – проміжне значення кількості координат у гілці,

xx – проміжне значення номера значущої позиції вектора *R*.

VN – вектор номерів гілок, що перетинають попередній рядок, де *vn[in]* – окремі компоненти вектора з індексом *in*, *++in* – операція перед індексацією, *in++* – операція пост індексації,

V_N – вектор номерів гілок, що перетинають поточний рядок, де *v_n[i_n]* – окремі компоненти вектора з індексом *i_n*.

Таблиця 2. Таблиця переходів автомата з відповідними діями

Поточний стан та його код (*stan*)	Код входу (*vxid*)	Вхідна ознака	Наступний стан та його код	Дії при зміні стану
(0, 0), 0	0	----	(0, 0), 0	--------------
	1	1	(0, 1), 1	i_n=i_n+1, xx=j.
	2	1	(0, 1), 2	in=in+1.
	3	1, 1	(1, 1), 4	*** nn=vn[++in], v_n[++i_n]=nn, zz=z[nn], x[nn][zz]=j, z[nn]=zz+1.
(0, 1), 1	0	----	(0, 1), 1	--------------
	1	0	(0, 0), 0	* v_n[i_n++]=ss, n[ss]=ss, p[ss]=1, y[ss]=i, zz=0, x[ss][zz]=xx, z[ss++]=1, v_n[i_n]=ss, n[ss]=ss, p[ss]=0, y[ss]=i, zz=0, x[ss][zz]=j, z[ss++] =1.
	2	1	(1, 1), 3	*** nn=vn[++in], v_n[i_n]=nn, zz=z[nn], x[nn][zz]=xx, z[nn]=zz+1.
	3	1, 0	(1, 0), 6	*** nn=vn[++in], v_n[i_n++]=nn, zz=z[nn], x[nn][zz]=xx, z[nn]=zz+1.
(0, 1), 2	0	----	(0, 1), 2	--------------
	1	1	(1, 1), 4	*** nn=vn[in], v_n[++i_n]=nn, zz=z[nn], x[nn][zz]=j, z[nn]=zz+1.
	2	0	(0, 0), 5	** n_min=min{vn[in+1], vn[in]} , n_max=max{vn[in+1], vn[in++]}, n_p[n_min]= n_max, n_p[n_max]= n_min.
	3	1, 0	(1, 0), 7	*** nn=vn[in++], v_n[++i_n]=nn, zz=z[nn], x[nn][zz]=j, z[nn]=zz+1.
(1, 1), 3	0	----	(1, 1), 3	--------------
	1	0	(1, 0), 6	i_n=i_n+1, xx=j.
	2	0	(1, 0), 7	in=in+1/
	3	0, 0	(0, 0), 0	*** nn=vn[++in], v_n[++i_n]=nn, zz=z[nn], x[nn][zz]=j, z[nn]=zz+1.
(1, 1), 4	0	----	(1, 1), 4	--------------
	1	0	(1, 0), 6	i_n=i_n+1, xx=j.
	2	0	(1, 0), 7	in=in+1.
	3	0, 0	(0, 0), 0	*** nn=vn[++in], v_n[++i_n]=nn, zz=z[nn], x[nn][zz]=j, z[nn]=zz+1.
(0, 0), 5	0	----	(0, 0), 5	--------------
	1	1	(0, 1), 1	i_n=in+1, xx=j.
	2	1	(0, 1), 2	in=in+1.
	3	1, 1	(1, 1), 4	*** nn=vn[++in], v_n[++i_n]=nn, zz=z[nn], x[nn][zz]=j, z[nn]=zz+1.
(1, 0), 6	0	----	(1, 0), 6	--------------
	1	1	(1, 1), 4	* v_n[i_n++]=ss, n[ss]=ss, p[ss]=0, y[ss]=i, zz=0, x[ss][zz]=xx, z[ss++]=1, v_n[i_n]=ss, n[ss]=ss, p[ss]=1, y[ss]=i, zz=0, x[ss][zz]=j, z[ss++] =1.
	2	0	(0, 0), 5	nn=vn[++in], v_n[i_n]=nn, zz=z[nn], x[nn][zz]=xx, z[nn]=zz+1.
	3	1, 0	(1, 0), 7	** v_n[i_n++]=ss, n[ss]=ss, p[ss]=0, y[ss]=i, zz=0, x[ss][zz]=xx, z[ss++]=1, v_n[i_n]=ss, n[ss]=ss, p[ss]=1, y[ss]=i, zz=0, x[ss][zz]=j, z[ss++] =1, in=in+1.
(1, 0), 7	0	----	(1, 0), 7	--------------
	1	0	(0, 0), 0	*** nn=vn[in], v_n[++i_n]=nn, zz=z[nn], x[nn][zz]=j, z[nn]=zz+1.
	2	1	(1, 1), 3	** n_min=min{vn[in+1], vn[in]} , n_max=max{vn[in+1], vn[in++]}, n_p[n_min]=n_max, n_p[n_max]= n_min.
	3	1, 0	(1, 0), 6	** n_min=min{vn[in+1], vn[in]} , n_max =max{vn[in+1], vn[in++]}, n_p[n_min]=n_max, n_p[n_max]=n_min, i_n=i_n+1, xx=j.

У таблицях 3 і 4 представлені результати обробки тестового зображення при формуванні таблиці векторів послідовно за рядками без деталізації процесу аналізу рядків. У таблиці 3 відображені результуючі значення векторів *T, R, VN, V_N* та коментар до ситуацій, що виникли при обробці кожного з рядків зображення. У таблиці

4 представлено вміст таблиці контурів після аналізу кожного рядка, з чого достатньо легко можна дослідити його послідовну зміну.

Таблиця 3. Результат обробки тестового зображення за рядками

i	T_i , R_i , *j=0-10*	*VN / V_N*	Коментар
3	*T* =00000000000 *R* =00000000000	_,	Дій не відбувається
4	*T* =00001000100 *R* =00001000100	_,0,1	Зародження пари гілок з номерами 0, 1
5	*T* =00010100110 *R* =00012100310	_,0,2,3,1	Перехід гілок 0, 1 з рядка в рядок Зародження пари гілок з номерами 2, 3
6	*T* =00101000101 *R* =00121200321	_,0,2,3,1	Перехід гілок 0, 2, 3, 1 з рядка в рядок
7	*T* =00101101101 *R* =00303101303	_,0,2,4,5,3,1	Перехід гілок 0, 2, 3, 1 з рядка в рядок Зародження пари гілок з номерами 4, 5
8	*T* =00010011100 *R* =00212213302	_,0,6,7,1	Перехід гілок 0, 1 з рядка в рядок З'єднання кінців гілок 2 і 4 Зародження пари гілок з номерами 6, 7 З'єднання кінців гілок 5 і 3
9	*T* =00000000000 *R* =00020022200	_,	З'єднання кінців гілок 0 і 6, 7 і 1

Таблиця 4. Вміст таблиці контурів у процесі обробки тестового зображення за рядками

n	*p*	*y*	*z*	*x*	*n_p*		*n*	*p*	*y*	*z*	*x*	*n_p*
i=4							*i=8*					
0	1	4	1	4			0	1	4	5	4, 3, 2, 2, 3	
1	0	4	1	8			1	0	4	5	8, 9, 10, 10, 8	
i=5							2	0	5	3	5, 4, 4	4
0	1	4	2	4, 3			3	1	5	3	8, 8, 8	5
1	0	4	2	8, 9			4	1	7	1	5	2
2	0	5	1	5			5	0	7	1	7	3
3	1	5	1	8			6	0	8	1	6	
i=6							7	1	8	1	7	
0	1	4	3	4, 3, 2			*i=9*					
1	0	4	3	8, 9, 10			0	1	4	5	4, 3, 2, 2, 3	6
2	0	5	2	5, 4			1	0	4	5	8, 9, 10, 10, 8	7
3	1	5	2	8, 8			2	0	5	3	5, 4, 4	4
i=7							3	1	5	3	8, 8, 8	5
0	1	4	4	4, 3, 2, 2			4	1	7	1	5	2
1	0	4	4	8, 9, 10, 10			5	0	7	1	7	3
2	0	5	3	5, 4, 4			6	0	8	1	6	0
3	1	5	3	8, 8, 8			7	1	8	1	7	1
4	1	7	1	5								
5	0	7	1	7								

У таблиці 5, як приклад, надано детальній опис дій та значення змінних у процесі послідовного аналізу 8-ого рядка тестового зображення. У стовпчику, позначеному *j,* розташовані номери позицій вектора *R*, починаючи з першої значущої, а в стовпчику, позначеному *vxid,* значення цього вектора відповідно позицій. У стовпчиках, позначених *stan, ss, in, i_n, xx, V_N* – значення цих змінних, які вони приймають по закінченні аналізу відповідної позиції. Вектор *V_N* послідовно заповнюється номерами гілок, що перейшли з попереднього, або зародилися в поточному рядку. Опис дій надано у вигляді переліку операцій над змінними, скопійованими з таблиці 2, відповідно до значень *stan, vxid* та тих же операцій, але з підстановкою чисельних

даних. Результат цих операцій, що змінили вміст таблиці контурів, можна порівняти для контролю з відповідними значеннями таблиці 4 для *i*, що дорівнює 8.

Таблиця 5. Результати аналізу 8-ого рядка зображення

(початок аналізу: *i=8*; *VN=_,0,2,4,5,3,1; ss=6; in=0*, *i_n=0*).

j	*stan*	*vxid*	*ss, in, i_n, xx* *V_N*	Дії при аналізі компонентів вектора *R*, сформованого для поточного рядка за його окремими значущими компонентами.
2	0	2	6, 0, 0, _ V_N=_	stan=2, in=in+1 in=0+1
3	2	1	6, 1, 0, _ V_N=_,0	stan =4,nn=vn[in], v_n[++i_n]=nn, zz=z[nn], x[nn][zz]=j, z[nn]=zz+1. nn=0, v_n[1]=0, zz=z[0]=4, x[0][4]=3, z[0]=zz+1=5.
4	4	2	6, 1, 1, _ V_N=_,0	stan =7, in=in+1=2.
5	7	2	6, 2, 1, _ V_N=_,0	stan =3, n_min=min{vn[in+1], vn[in]} , n_max=max{vn[in+1], vn[in++]}, n_p[n_min]=n_max, n_p[n_max]= n_min, n_min=min{vn[3], vn[2]}=2 , n_max=max{vn[3], vn[2++]}=4, n_p[2]=4, n_p[4]= 2.
6	3	1	6, 3, 1, _ V_N=_,0	stan =6, i_n=i_n+1, xx=j. i_n=1+1, xx=6.
7	6	3	6, 3, 2, 6 V_N=_,0,6,7	stan =7, v_n[i_n++]=ss, n[ss]=ss, p[ss]=0, y[ss]=i, zz=0, x[ss][zz]=xx, z[ss++] =1, v_n[i_n]=ss, n[ss]=ss, p[ss]=1, y[ss]=i, zz=0, x[ss][zz]=j, z[ss++] =1, in=in+1. v_n[2++]=6, n[6]=6, p[6]=0, y[0]=8, zz=0, x[6][0]=6, z[6++] =1, v_n[3]=7, n[7]=7, p[7]=1, y[7]=7, zz=0, x[7][0]=7, z[7++] =1, in=3+1=4.
8	7	3	8, 4, 3, 6 V_N=_,0,6,7	stan =6, n_min=min{vn[in+1], vn[in]} , n_max =max{vn[in+1], vn[in++]}, n_p[n_min]=n_max, n_p[n_max]=n_min, i_n=i_n+1, xx=j n_min=min{vn[5],vn[4]}=3 , n_max =max{vn[5], vn[4++]}=5, n_p[3]=5, n_p[5]=3, i_n=3+1, xx=8.
9	6	0	8, 5, 4, 8 V_N=_,0,6,7	stan =6, *дій не відбувається.*
10	6	2	8, 5, 4, 8 V_N=_,0,6,7, 1	stan =5, nn=vn[++in], v_n[i_n]=nn, zz=z[nn], x[nn][zz]=xx, z[nn]=zz+1. nn=vn[++5]=1, v_n[4]=1, zz=z[1]=4, x[1][4]=8, z[1]=4+1.

Наступним етапом є об’єднання гілок в контури, тобто позначення гілок, що відносяться до контуру одного об’єкта номером контуру замість номера гілки та перерахунок координат гілок. Процедура буде такою.

Вводиться змінна лічильника контурів *k*. На початку процедури *k* дорівнює нулю, в кінці аналізу буде дорівнювати кількості контурів. У таблиці контурів скануються по вертикалі номери гілок. При зустрічі номера гілки, що більший або дорівнює *k* та враховуючи, що при зародженні пара гілок з’єднана своїми початками, починається послідовне слідкування за гілками, з’єднаними кінцями і початками доти, поки не повернемось до гілки, з якої почали. Номери з’єднаних між собою гілок змінюються на значення *k*. Збільшуємо *k* на одиницю і продовжуємо процес сканування і відслідковування ланцюжків гілок, з’єднаних у замкнені контури.

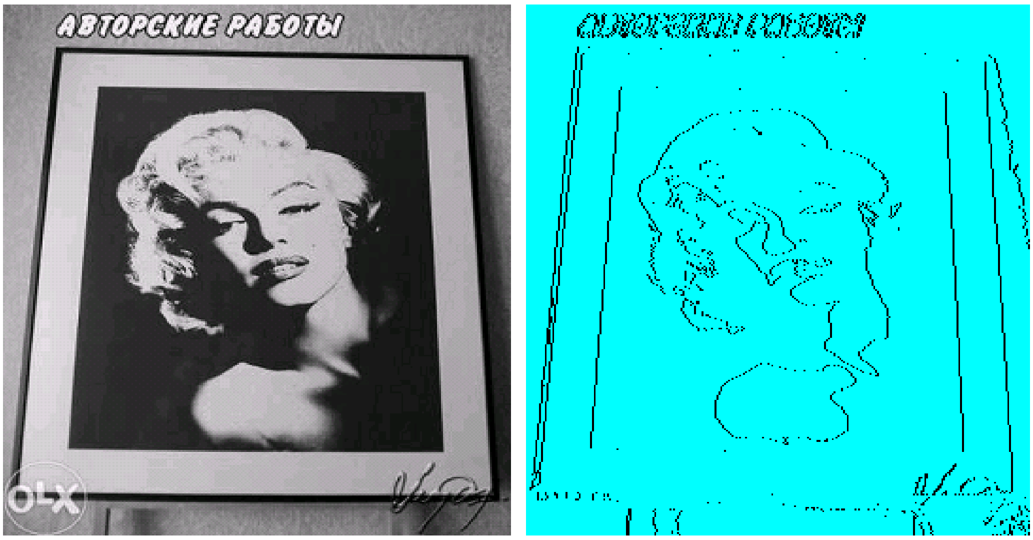
Загальна послідовність координат точок у контурі за гілками має наступний порядок. Для першої гілки береться послідовність координат від початку гілки до її кінця, для наступної, навпаки, від кінця – до початку і так далі, по черзі змінюючи напрям. Цей порядок відповідає обходу контуру проти годинникової стрілки.

Колір об’єкта, замкненого контуром (чорний або білий), визначає ознака першої гілки. Враховуючи умову, що контур утворюють граничні пікселі, які належать об’єкту, координати *x* всіх білих гілок чорного контуру, та координати *x* всіх чорних гілок білого контуру зменшують на одиницю. У таблиці 6 представлена фінальна таблиця гілок контурів після вказаних перетворень. У результаті отримані контури двох об’єктів, чорного і білого.

Таблиця 6. Фінальна таблиця гілок контурів після перетворень

n	*p*	*y*	*z*	*x*	*n_p*
0	1	4	5	4, 3, 2, 2, 3	6
0	0	4	5	7, 8, 9, 9, 7	7
1	0	5	3	5, 4, 4	4
1	1	5	3	7, 7, 7	5
1	1	7	1	4	2
1	0	7	1	7	3
0	0	8	1	5	0
0	1	8	1	7	1

На рисунку 2а представлене напівтонове зображення, а на рисунку 2б відображені точки контурів, координати яких увійшли до фінальної таблиці гілок контурів. Перед кодуванням контурів здійснена проста порогова бінаризація. Значення порога – 100, загальна кількість гілок в таблиці – 534, контурів – 132.



а б

Рис. 2. а – кольорове зображення, б - точки контурів, за координатами з таблиці гілок контурів

З рисунку 2б видно, що значна частина точок контурів об’єктів не відображена. Точніше, немає горизонтальних ліній, відображені тільки їх крайні точки. Але за потреби формування повного вектору координат точок контуру вся необхідна інформація для обчислення координат відсутніх точок в таблиці є. Для цього необхідно аналізувати різницю між сусідніми координатами *x* контуру. Абсолютне значення різниці вкаже на кількість відсутніх точок, а знак різниці та ознака гілки – на напрямок їх розташування. Ця процедура досить проста. Тому не будемо її детально розглядати, оскільки для обчислення моментів інерції бінарних об’єктів вона не потрібна.

Обчислення моментів інерції бінарних об’єктів, замкнених контурами з використанням векторних операцій.

Одними з параметрів, що характеризують форму об'єктів, є моменти інерції різних порядків. На їх основі для задач порівняння об'єктів за формою контурів можна

будувати моментні інваріанти, які не залежать від афінних перетворень: зсуву, повороту, зміни масштабу об'єкта та інших перетворень [2].

Нижче, для дискретного випадку, представлена загальна формула обчислення моментів *к*-ого порядку:

$$M_{j,k-j} = \sum_{i=1}^{N} B_i \cdot x_i^j \cdot y_i^{k-j}, \; j = (0, k),$$

де $M_{j,k-j}$ – моменти; B_i – значення яскравостей пікселів об'єкта; x_i, y_i – координати пікселів; N – кількість пікселів бінарного об'єкта.

Відповідно до прийнятої моделі, за об'єкт будемо вважати площу, замкнену контуром з однаковою яскравістю всіх пікселів, яку можна винести за знак суми і умовно прирівняти до одиниці.

На зображенні в межах одного рядка може перебувати кілька сегментів об'єкта. Тобто, об'єкт можна представляти як сукупність його сегментів у різних рядках, а момент деякого порядку цього об'єкта як суму моментів такого ж порядку всіх його сегментів. Отже, знаючи початок і кінець сегмента в конкретному рядку, можна визначити його момент $MS_{j,k-j}$ як:

$MS_{j,k-j} = y^{k-j} \cdot \sum_{i=1}^{n} x_i^j$, де y – номер рядка, а n – кількість пікселів у сегменті.

Отже, якщо поточний граничний піксель з координатами (x_i, y_i) є початком сегмента в рядку, то: $M_{j,k-j} = M_{j,k-j} - y_i^{k-j} \cdot \sum_{a=1}^{x_i - 1} a^j$,

а якщо кінцем тоді: $M_{j,k-j} = M_{j,k-j} + y_i^{k-j} \cdot \sum_{a=1}^{x_i} a^j$,

де $\sum_{a=1}^{x_i} a^j$ – сума членів арифметичної прогресії в степені j.

Нижче представлені вирази для сум членів арифметичної прогресії в степені 1, 2, 3. Спосіб виведення формул для степенів довільного порядку надано в [3].

$$\sum_{a=1}^{x_i} a^1 = \frac{x_i \cdot (x_i + 1)}{2}, \; \sum_{a=1}^{x_i} a^2 = \frac{x_i \cdot (x_i + 1)(2x_i + 1)}{6}, \; \sum_{a=1}^{x_i} a^3 = \frac{x_i^2 \cdot (x_i + 1)^2}{2}.$$

З процедури формування таблиці контурів є очевидним, що для чорного контуру точки чорних гілок є початками, а точки білих гілок – кінцями сегментів, і навпаки, для білого контуру точки білих гілок є початками, а точки чорних гілок – кінцями сегментів об'єкта, замкненого цими контурами. Наявність у таблиці контурів координат граничних точок контурів для кожного рядка є гарантією того, що в таблиці контурів обов'язково присутні початок і кінець усіх сегментів об'єктів.

Наведемо приклад розрахунку моменту другого порядку M_{11} для білого об'єкта тестового зображення послідовним способом та за запропонованим алгоритмом з використанням векторних операцій.

Послідовний спосіб (дивись рис.1):

$M_{11} = 5 \cdot (5 + 6 + 7) + 6 \cdot (4 + 5 + 6 + 7) + 7 \cdot (4 + 7) = 299$.

Результат за запропонованим алгоритмом представлений в таблиці 7.

Таблиця 7. Результат обчислень моменту M_{11} за запропонованим алгоритмом

Номер гілки	Ознака гілки p	Вектори Y, X, S, $\pm$ YxS	Значення векторів
2	0	$Y\{y_i\}$	5, 6, 7
		$X\{x_i\}$	5, 4, 4
		S	10, 6, 6
		-YxS	50, 36, 42
3	1	$Y\{y_i\}$	5, 6, 7
		$X\{x_i\}$	7, 7, 7
		S	28, 28, 28
		+YxS	140, 168, 196
4	1	$Y\{y_i\}$	7
		$X\{x_i\}$	4
		S	10
		+ YxS	70
5	0	$Y\{y_i\}$	7
		$X\{x_i\}$	7
		S	21
		-YxS	147
Вектор суми добутків для всіх гілок			13, 132, 154
Загальна сума			299

Y – вектор координат y_i, X – вектор координат x_i,

S – вектор сум $\sum_{a=1}^{x_i} a^1$ для чорної гілки та $\sum_{a=1}^{x_i-1} a^1$ для білої гілки,

YxS – вектор добутку компонентів Y та S.

Висновок

У результаті досліджень запропоновано алгоритм, який дозволяє на етапі кодування контурів та їх перетворення проводити обробку відеоданих, отриманих на попередніх етапах, не чекаючи вводу всього кадру зображення. Це можливо завдяки тому, що алгоритми попередніх етапів (фільтрації та бінаризації) також дозволяють отримувати результати в режимі суміщення введення і обробки відеоінформації.

Запропоновані алгоритми кодування і обчислення моментів інерції за допомогою векторних операцій, можуть значно прискорити обчислювальний процес на сучасних обчислювальних пристроях, або ннададуть поштовх для створення нових високоефективних паралельних процесорів. Це обумовлено наявністю на ринку процесорів з апаратною підтримкою векторних операцій з фіксованою і плаваючою точкою (наприклад, процесори фірми ARM), а також наявністю високоінтегрованих ПЛІС (наприклад, FPGA фірми Xilinx), на яких можна реалізувати векторні процесори для паралельної обробки декількох тисяч даних.

У роботі на конкретних прикладах продемонстровано принципи використання векторних операцій при кодуванні контурів та обчисленні моментів інерції за запропонованими алгоритмами. На основі цих принципів можуть бути розроблені і інші більш ефективні алгоритми.

Література

1. Абламейко С. В. Обработка изображений: технология, методы, применение: Учеб. пособие. / С. В. Абламейко, Д. М. Лагуновский. – М.: Амалфей, 2000. – 304 с.
2. Глумов Н.И. Построение и применение моментных инвариантов для обработки изображений в скользящем окне / Н.И. Глумов // Компьютерная оптика. – 1995. – № 14. – С. 46-54.
3. Формулы суммы первой степени, квадратов, кубов первых *n* натуральных чисел [Електр. ресурс] // «Математика – это просто!» – некоммерческий, обучающий сайт, 2009 – 2013, – Режим доступа: http://easymath.com.ua/suma_pervoj_stepeni_kvadratov_kubov_pervyh_n_naruralnyh_chisel.php – Загл. с экрана. (27.03.2017).

Literatura

1. Ablameiko S. V. Obrabotka izobradzeni: tehnologiya, metodi, primenenie: Ucheb. Posobie. / S. V. Ablameiko, D. M. Lagunovski. – M: Amalffei, 2000. – 304 s.
2. Glumov N.I. Postroenie i primenenie momentnih invariantov dlya obrabotki izobradzeni v skolzyachem okne / N.I. Glumov // Komputernaya optika. – 1995. – № 14. – S. 46-54.
3. Formuli summi pervoi stepeni, kvadratov, kubov pervih *n* naturalnih chisel [Elektr. Resurs] // «Matematika eto prosto! – nekommercheski, obuchayuchi sait, 2009 – 2013 – Rezhym dostupu: **http://easymath.com.ua/suma_pervoj_stepeni_kvadratov_kubov_pervyh_n_naruralnyh_chisel.php** – Zagl. s ekrana (27.03.2017).

RESUME

P.Yu. Sabelnikov

Parallel encoding the contours of objects in the binary image and the calculation of its moments of inertia

It is enough important in image processing, especially by embedded devices, designed to work in real-time, is the problem of speeding up calculations. Much of the procedure, such as pre-processing images, is reduced to a large number of similar operations on 1-2 byte integers. This feature allows in a natural way to parallelize the processing process through the use of vector operations. Manufacturers of processors have long proposed an efficient solution for increasing the performance when working with such data. SIMD (Single instruction, multiple data) are sets of vector instructions that allow processing a large amount of data in one operation. For example, on the x86 platform (Intel company) there are a large number of SIMD extensions: MMX, 3DNow !, SSE, SSE2, SSE3, SSEE3, SSE4.1, SSE4.2, AVX and AVX2. Subsystem of vector operations for processing numbers with fixed and floating point (Neon extension) is also implemented in the ARM processors.

However, a significant number of areas in image processing do not allow us to directly apply this approach. To improve the efficiency of video data processing in the SIMD mode, it is necessary to develop new or modify existing methods and algorithms using vector subsystems.

In this paper we propose algorithms for coding the contours of objects in binary images and calculating their moments of inertia, which can greatly speed up the calculation process by combining the input and processing of image data using vector operations.

The procedure of sequential selection and encoding of contours assumes the presence of a complete frame of image in the memory of the computing device. In order to reduce the time lag of processing results, save memory resources and improve performance, it is advisable to perform calculations in parallel with the introduction of the image. This goal is met by the method of selection and coding the contours of objects of binary video images, based on sequential scanning of a part of the image lines.

The algorithm of coding contours is based on the sequential input of image lines and analysis of the order of following in these lines the black and white segments of the image. To implement the algorithm in the memory of the processing device, it is necessary to store only the entered and the previous line of the image.

One of the parameters that characterize the shape of objects are moments of inertia of various orders. On their basis for the task of comparing objects on shape of contours, one can build moment invariants that do not depend on affine transformations: of shift, rotation, scaling of the object, and other transformations.

The proposed algorithms for coding contours and calculating the moments of inertia of binary objects allow the use of vector operations. Currently, in many processors are implemented hardware support of vector operations on numbers with fixed and floating point. In the market there are also highly integrated the FPGA, such as Xilinx company, allowing to build a vector processors for parallel processing of several thousand numbers.

In the article on the specific examples are demonstrated how to use vector operations for the implementation of the proposed algorithms. And other more efficient algorithms can be developed on the basis of these principles.

Надійшла до редакції 23.09.2016

